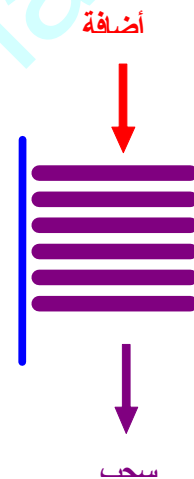


QUEUES

A *queue* is simply a waiting line that grows by adding elements to its end and shrinks by taking elements from its front. Unlike a stack, a queue is a structure in which both ends are used: one for adding new elements and one for removing them. Therefore, the last element has to wait until all elements preceding it on the queue are removed. A queue is an FIFO structure: first in/first out.



Clear()—Clear the queue.

isEmpty()—Check to see if the queue is empty.

enqueue(*el*)—Put the element *el* at the end of the queue.

outqueue()—Take the first element from the queue.

```

// queue.cpp : by Dr. Magdi El-Sharkawy.
#include "stdafx.h"
#include <iostream>
using namespace std;
//----- class IntNode for creating new node -----//
class IntNode {
public :
    IntNode(int el, IntNode *ptr = 0) {info = el; next = ptr;}
    int info;
    IntNode *next;
};
//----- class IntLList for dealing with nodes -----//
class queue {
public:
    queue() {head = tail =0; }
    bool isempty();
    void enqueue(int);
    int outqueue();
    void Displayqueue();
    void clear();
private:
    IntNode *head, *tail;
};

bool queue::isempty()
{
    if (head==0 && tail ==0)
        return 1;
    else
        return 0;
}

void queue::enqueue(int data)
{
    IntNode *newnode;
    newnode = new IntNode(data,0);
    if(tail!=0)
    {
        tail->next = newnode;
        tail = newnode;
    }
    else
        head=tail= newnode;
}

```

```

int queue::outqueue()
{
    int val=0;
    if(head!=0)
    {
        val = head->info;
        head = head->next;
    }
    if(head==0)
        tail=head;
    return val;
}

void queue::clear()
{
    head = tail =0;
}

void queue::Displayqueue()
{
    IntNode *current;
    current = head;
    while(current != tail->next)
    {
        cout << current->info << "    " << current << "\n";
        current=current->next;
    }
    cout << "-----" << "\n";
}

void main()
{
    queue mag;
    if(mag.isEmpty()) cout << "queue is empty \n";
    else cout << "queue is not empty \n";
    mag.enqueue(50);
    mag.enqueue(90);
    mag.enqueue(60);
    mag.Displayqueue();
    cout << mag.outqueue() << "    popped out \n ----- \n";
    mag.Displayqueue();
    mag.enqueue(66);
    mag.enqueue(45);
    mag.enqueue(38);
    mag.Displayqueue();
}

```

```

    cout << mag.outqueue() << "    popped out \n ----- \n";
    mag.Displayqueue();
    mag.enqueue(50);
    mag.Displayqueue();
    mag.clear();
    if(mag.isempty()) cout << "queue is empty \n";
        else cout << "queue is not empty \n";
}

```

```

queue is empty
50    00091990
90    000919C8
60    00091A00
-----
50    popped out
-----
90    000919C8
60    00091A00
-----
90    000919C8
60    00091A00
66    000B1F50
45    000B1F88
38    000B1FC0
-----
90    popped out
-----
60    00091A00
66    000B1F50
45    000B1F88
38    000B1FC0
-----
60    00091A00
66    000B1F50
45    000B1F88
38    000B1FC0
50    000B1D40
-----
queue is empty

```